

Capítulo 1: Tipos de datos y Operadores

Programación 2

Comentarios en Python

Existen dos tipos de comentarios:

Comentarios de una sola línea se realizan con el numeral.

#Ejemplo de comentario de una sola línea

Comentarios multilineas se realizan con comillas triples.

```
"""Este es un comentario  
multilínea utilizando  
triple comillas"""
```

Operadores aritméticos

| | | |
|------------------------|----------------------|---|
| <code>1 + 1</code> | <code># 2</code> | Suma |
| <code>8 - 1</code> | <code># 7</code> | Resta |
| <code>10 * 2</code> | <code># 20</code> | Multiplicación |
| <code>5 ** 2</code> | <code># 25</code> | Potencia |
| <code>pow(5, 2)</code> | <code># 25</code> | Potencia |
| <code>25 ** 0,5</code> | <code># 5</code> | Raíz con potencia fraccionaria |
| <code>35 / 5</code> | <code># 7.0</code> | División (devuelve float) |
| <code>35 / 0</code> | <code># Error</code> | |
| <code>34 // 5</code> | <code># 6</code> | División entera (trunca el cociente) |
| <code>35 % 6</code> | <code># 5</code> | Operador módulo (resto) |
| <code>3 * 2.0</code> | <code># 6.0</code> | Si uno de los operandos es float, el resultado es float |

Operadores lógicos

Valores “boolean” primitivos:

- True
- False

Operadores booleanos nativos:

not:

- | | |
|-----------------|---------|
| • not True | # False |
| • not False | # True |
| • True and True | # True |

and:

- | | |
|-------------------|---------|
| • True and True | # True |
| • True and False | # False |
| • False and True | # False |
| • False and False | # False |

or:

- | | |
|------------------|---------|
| • True or True | # True |
| • True or False | # True |
| • False or True | # True |
| • False or False | # False |

Operadores lógicos

Cortocircuitos por defecto:

- True and False and 1 / 0 # False
- True and True and 1 / 0 # Error
- False or True or 1 / 0 # True
- False or False or 1 / 0 # Error

Operadores de comparación

Operadores básicos:

1 == 1 # True

1 != 1 # False

1 < 10 # True

1 > 10 # False

2 <= 2 # True

2 >= 2 # True

Operadores de comparación

Comparación combinada:

`1 < 2 < 3` `# True`

`1 < 3 < 2` `# False`

`1 < 0 < 1 / 0` `# False` (por cortocircuito)

`1 <= 2 <= 4 <= 6` `# True`

`1 != 2 <= 4 != 6` `# True`

Nota: la longitud de la combinación puede ser indefinida

Tipos de datos: Cadena de caracteres (strings)

Los strings se crean con comillas simples o dobles.

`"Esto es un string"`

`'Esto es un string'`

Los strings se pueden concatenar

`"hola " + "mundo"` `# hola mundo`

`"hola " "mundo"` `# hola mundo`
 (concatenación automática)

Los strings funcionan también como listas

`"Esto es un string"[0]` `# 'E' (String como lista)`

Tipos de datos: Cadena de caracteres (strings)

Formateo de strings: Se denomina formateo de strings a la posibilidad de introducir variables dentro de un string.

```
nombre = "Ezequiel"
```

```
precio = 12.50
```

```
descuento = 0.8
```

```
comida = "lasagna"
```

```
"{} debe pagar ${}".format(nombre,precio)
```

```
# Ezequiel debe pagar $12.50
```

```
"{} debe pagar ${}".format(nombre,precio*descuento)
```

```
# Ezequiel debe pagar $10.00
```

```
"{} debe pagar ${}".format(precio=precio, nombre="Bob")
```

```
# Bob debe pagar $12.50 (En este caso no importa el orden)
```

Formateo de strings con f-Strings:

```
f'{nombre} quiere comer {comida}'
```

```
#Ezequiel quiere comer lasagna
```

Objeto None

Los lenguajes en general tienen un objeto particular que definen una no salida.

Java = null

C = void

Python = None

Nota: None NO significa Falso ni Indefinido.

True is None # False

False is None # False

None is None # True

Valores interpretados como booleanos

En caso de los enteros:

`bool(0)` `#False` (El 0 se interpreta como False)

`bool(1)` `#True` (El 1 se interpreta como True)

En el caso de los strings:

`bool("")` `#False` (Los strings vacíos se interpretan
como False)

`bool("a")` `#True` (String con contenido True)

En el caso de las listas:

`bool([])` `#False` (Listas vacías False)

`bool([3])` `#True` (Listas con contenido True)

Valores interpretados como booleanos

Interpretación booleana de tipos no booleanos.

| | |
|----------------------|---------------------|
| <code>not "1"</code> | <code>#False</code> |
| <code>not []</code> | <code>#True</code> |

El resultado NO SIEMPRE va a ser True o False. En el caso de tener operaciones 'and' o de 'or' el resultado NUNCA va a ser True o False.

Regla: El último elemento evaluado que fue True incluyendo cortocircuito.

Valores interpretados como booleanos

Regla: El último elemento evaluado que fue True incluyendo cortocircuito.

and:

- 1 and [3] # [3] True and True
- [3] and 1 # 1 True and True
- [3] and 0 # 0 True and False
- 0 and [3] # 0 False and True
- 0 and "" # 0 False and False
- "" and 0 # "" False and False

or:

- 1 or [3] # 1 True or True
- [3] or 1 # [3] True or True
- 1 or [] # 1 True or False
- [] or 1 # 1 False or True
- 0 or [] # [] False or False
- [] or 0 # 0 False or False

Ejemplo: Quiero saber si un parámetro enviado es vacío porque en ese caso le aplico un valor por defecto
`args = [] or [1,2]`

Conversiones numéricas de base

Decimal:

| | |
|------------------------|-------------------|
| <code>str(10)</code> | <code># 10</code> |
| <code>int("10")</code> | <code># 10</code> |

Binario

| | |
|-------------------------------|-------------------------|
| <code>bin(10)</code> | <code># '0b1010'</code> |
| <code>int("0b1010", 2)</code> | <code># 10</code> |
| <code>int("1010", 2)</code> | <code># 10</code> |

Octal

| | |
|-----------------------------|-----------------------|
| <code>oct(10)</code> | <code># '0o12'</code> |
| <code>int("0o12", 8)</code> | <code># 10</code> |
| <code>int("12", 8)</code> | <code># 10</code> |

Hexadecimal

| | |
|-----------------------------|---------------------|
| <code>hex(10)</code> | <code>#'0xa'</code> |
| <code>int("0xa", 16)</code> | <code># 10</code> |
| <code>int("0xA", 16)</code> | <code># 10</code> |
| <code>int("a", 16)</code> | <code># 10</code> |
| <code>int("A", 16)</code> | <code># 10</code> |

Conversiones de string

En Python soporta Unicode (como estandar de codificacion de caracteres).

| | |
|----------------------|--------------------|
| <code>chr(65)</code> | <code># "A"</code> |
|----------------------|--------------------|

| | |
|-----------------------|--------------------|
| <code>chr(191)</code> | <code># "¿"</code> |
|-----------------------|--------------------|

| | |
|-----------------------|-------------------|
| <code>ord("A")</code> | <code># 65</code> |
|-----------------------|-------------------|

| | |
|-----------------------|--------------------|
| <code>ord("¿")</code> | <code># 191</code> |
|-----------------------|--------------------|